# Domain adaptive bootstrapping for named entity recognition

**Dan Wu**[1], **Wee Sun Lee**[2], **Nan Ye**[2]
[1]Singapore MIT Alliance
[2]Department of Computer Science
National University of Singapore
{dwu@,leews@comp,g0701171@}nus.edu.sg

**Hai Leong Chieu**
DSO National Laboratories
chaileon@dso.org.sg

## Abstract

Bootstrapping is the process of improving the performance of a trained classifier by iteratively adding data that is labeled by the classifier itself to the training set, and retraining the classifier. It is often used in situations where labeled training data is scarce but unlabeled data is abundant. In this paper, we consider the problem of domain adaptation: the situation where training data may not be scarce, but belongs to a different domain from the target application domain. As the distribution of unlabeled data is different from the training data, standard bootstrapping often has difficulty selecting informative data to add to the training set. We propose an effective domain adaptive bootstrapping algorithm that selects unlabeled target domain data that are *informative about the target domain* and *easy to automatically label correctly*. We call these instances *bridges*, as they are used to bridge the source domain to the target domain. We show that the method outperforms supervised, transductive and bootstrapping algorithms on the named entity recognition task.

## 1 Introduction

Most recent researches on natural language processing (NLP) problems are based on machine learning algorithms. High performance can often be achieved if the system is trained and tested on data from the same domain. However, the performance of NLP systems often degrades badly when the test data is drawn from a source that is different from the labeled data used to train the system. For named entity recognition (NER), for example, Ciaramita and Altun (2005) reported that a system trained on a labeled Reuters corpus achieved an F-measure of 91% on a Reuters test set, but only 64% on a Wall Street Journal test set.

The task of adapting a system trained on one domain (called the *source* domain) to a new domain (called the *target* domain) is called domain adaptation. In domain adaptation, it is generally assumed that we have labeled data in the source domain while labeled data may or may not be available in the target domain. Previous work in domain adaptation can be classified into two categories: [S+T+], where a small, labeled target domain data is available, e.g. (Blitzer et al., 2006; Jiang and Zhai, 2007; Daumé III, 2007; Finkel and Manning, 2009), or [S+T-], where no labeled target domain data is available, e.g. (Blitzer et al., 2006; Jiang and Zhai, 2007). In both cases, and especially for [S+T-], domain adaptation can leverage on large amounts of unlabeled data in the target domain. In practice, it is often unreasonable to expect labeled data for every new domain that we come across, such as blogs, emails, a different newspaper agency, or simply articles from a different topic or period in time. Thus although [S+T+] is easier to handle, [S+T-] is of higher practical importance.

In this paper, we propose a domain adaptive bootstrapping (DAB) approach to tackle the domain adaptation problem under the setting [S+T-]. Bootstrapping is an iterative process that uses a trained classifier to label and select unlabeled instances to add to the training set for retraining the classifier. It is often used when labeled training data is scarce but unlabeled data is abundant. In contrast, for domain adaptation problems, we may have a lot of training data but the target application domain has a different data distribution. Standard bootstrapping usually selects instances that are most confidently labeled from the unlabeled data. In domain adaptation situations, usually the most confidently labeled instances are the ones that are most similar to the source domain instances - these instances tend to contain very little information about the target domain. For domain adaptive bootstrapping, we propose a selection criterion that selects instances that are informative and easy to automatically label correctly. In addition, we propose a criterion for stopping the process of bootstrapping before it adds uninformative and incorrectly labeled instances that can reduce performance.

Our approach leverages on instances in the target domain called *bridges*. These instances contain domain-independent features, as well as features specific to the target domain. As they contain domain-independent features, they can be classified correctly by classifiers trained on the source domain labeled data. We argue that these instances act as a bridge between the source and the target domain. We show that, on the NER task, DAB outperforms supervised, transductive and standard bootstrapping algorithms, as well as a bootstrapping variant, called *balanced* bootstrapping (Jiang and Zhai, 2007), that has recently been proposed for domain adaptation.

## 2 Related work

One general class of approaches to domain adaptation is to consider that the instances from the source and the target domain are drawn from different distributions. Bickel et al. (Bickel et al., 2007) discriminatively learns a scaling factor for source domain training data, so as to adapt the source domain data distribution to resemble the target domain data distribution, under the [S+T-] setting. Daume III and Marcu (Daumé III and Marcu, 2006) considers that the data distribution is a mixture distribution over general, source domain and target domain data. They learn the underlying mixture distribution using the conditional expectation maximization algorithm, under the [S+T+] setting. Jiang and Zhai (2007) proposed an instance re-weighting framework that handles both the [S+T+] and [S+T-] settings. For [S+T-], the resulting algorithm is a *balanced* bootstrapping algorithm, which was shown to outperform the *standard* bootstrapping algorithm. In this paper, we assume the [S+T-] settings, and we show that the approach proposed in this paper, domain adaptive bootstrapping (DAB), outperforms the balanced bootstrapping algorithm on NER.

Another class of approaches to domain adaptation is feature-based. Daume III (Daumé III, 2007) divided features into three classes: domain-independent features, source-domain features and target-domain features. He assumed the existence of training data in the target-domain (under the setting [S+T+]), so that the three classes of features can be jointly trained using source and target domain labeled data. This cannot be done in the setting [S+T-], where no training data is available in the target domain. Using a different approach, Blitzer et al. (2006) induces correspondences between feature spaces in different domains, by detecting pivot features. Pivot features are features that occur frequently and behave similarly in different domains. Pivot features are used to put domain-specific features in correspondence. In this paper, instead of pivot features, we attempt to leverage on pivot instances that we call *bridges*, which are instances that bridge the source and target domain. This will be illustrated in Section 3.

It is generally recognized that adding informative and correctly labeled instances is more useful for learning. Active learning queries the user for labels of most informative or relevant instances. Active learning, which has been applied to the problem of NER in (Shen et al., 2004), is used in situations where a large amount of unlabeled data exists and data labeling is expensive. It has also been applied to the problem of domain adaptation for word sense disambiguation in (Chan and Ng, 2007). However, active learning requires human intervention. Here, we want to achieve the same goal without human intervention.

## 3 Bootstrapping for domain adaptation

We first define the notations used for domain adaptation in the [S+T-] setting. A set of training data $D_S = \{x_i, y_i\}_{1 \leq i \leq |D_S|}$ is given in the source domain, where the notation $|X|$ denotes the size of a set $X$. Each instance $x_i$ in $D_S$ has been manually annotated with a label, $y_i$, from a given set of labels $Y$. The objective of domain adaptation is to label a set of unlabeled data, $D_T = \{x_i\}_{1 \leq i \leq |D_T|}$ with labels from $Y$. A machine learning algorithm will take a labeled data set (for e.g. $D_S$) and outputs a *classifier*, which can then be used to classify unlabeled data, i.e. assign labels to unlabeled instances.

A special class of machine learning algorithms, called *transductive* learning algorithms, is able to take the unlabeled data $D_T$ into account during the learning process (see e.g. (Joachims, 1999)).

However, such algorithms do not take into account the shift in domain of the test data. Jiang and Zhai (2007) recently proposed an instance re-weighting framework to take domain shift into account. For [S+T-], the resulting algorithm is a *balanced* bootstrapping algorithm, which we describe below.

## 3.1 Standard and balanced bootstrapping

We define a general bootstrapping algorithm in Algorithm 1. The algorithm can be applied to any machine learning algorithm that allows training instances to be weighted, and that gives confidence scores for the labels when used to classify test data. The bootstrapping procedure iteratively improves the performance of a classifier $SC_t$ over a number of iterations. In Algorithm 1, we have left a number of parameters unspecified. These parameters are (1) the *selection-criterion* for instances to be added to the training data, (2) the *termination-criterion* for the bootstrapping process, and (3) the weights $(w^S, w^T)$ given to the labeled and bootstrapped training sets.

**Standard bootstrapping:** (Jiang and Zhai, 2007) the *selection-criterion* is based on selecting the top $k$ most-confidently labeled instances in $R_t$. The weight $w_t^S$ is equal to $w_t^T$. The value of $k$ is a parameter for the bootstrapping algorithm.

**Balanced bootstrapping:** (Jiang and Zhai, 2007) the *selection-criterion* is still based on selecting the top $k$ most-confidently labeled instances in $R_t$. Balanced bootstrapping was formulated for domain adaptation, and hence they set the weights to satisfy the ratio $\frac{w_t^S}{w_t^T} = \frac{|T_t|}{|D_S|}$. This allows the small amount of target data added, $T_t$, to have an equal weight to the large source domain training set $D_S$.

In this paper, we formulate a selection-criterion and a termination-criterion which are better than those used in standard and balanced bootstrapping. Regarding the selection-criterion, standard and balanced bootstrapping both select instances which are confidently labeled by $SC_t$ to be used for training $SC_{t+1}$, in the hope of avoiding using wrongly labeled data in bootstrapping. However, instances that are already confidently labeled by $SC_t$ may not contain sufficient information which is not in $D_S$, and using them to train $SC_{t+1}$ may result in $SC_{t+1}$ performing similarly to $SC_t$. This motivates us to select samples which are both *informative* and *easy to automatically label correctly*. Regarding the *termination-criterion*, which is not mentioned in the paper (Jiang and Zhai, 2007), we assume that bootstrapping is simply run for either a single iteration, or a small and fixed number of iterations. However, it is known that such simple criterion may result in stopping too early or too late, leading to sub-optimal performance. We propose a more effective termination-criterion here.

**Algorithm 1** Bootstrapping algorithm

```
Input: labeled data $D_S$, test data $D_T$ and a machine learning algorithm.
Output: the predicted labels of the set $D_T$.
Set $T_0 = \emptyset$, $R_0 = D_T$, and $t = 0$
Repeat
    1. learn a classifier $SC_t$ with $(D_S, T_t)$ with weights $(w_t^S, w_t^T)$
    2. label the set $R_t$ with $SC_t$
    3. select $S_t \subseteq R_t$ based on selection-criterion
    4. $T_{t+1} = T_t \cup S_t$, and $R_{t+1} = R_t \setminus S_t$.
Until termination-criterion
Output the predicted labels of $D_T$ by $SC_t$.
```

## 3.2 Domain adaptive bootstrapping (DAB)

Our selection-criterion relies on the observation that in domain adaptation, instances (from the source or the target domain) can be divided into three types according to their information content: *generalists* are instances that contain only domain-independent information and are present in all domains; *specialists* are instances containing only domain-specific information and are present only in their respective domains; *bridges* are instances containing both domain-independent and domain-specific information, also present only in their respective domains but are useful as a "bridge" between the source and the target domains.

The implication of the above observation is that when choosing unlabeled target domain data for bootstrapping, we should exploit the *bridges*, because the *generalists* are not likely to contain much information not in $D_S$ due to their domain-independence, and the *specialists* are difficult to be labeled correctly due to their domain-specificity. In contrast, the *bridges* are informative and easier to label correctly. Choosing confidently classified instances for bootstrapping, as in standard bootstrapping and balanced bootstrapping, is simple, but results in choosing mostly *generalists*, and is too conservative. We design a scoring function on instances, which has high value when the instance is informative and sufficiently likely to be correctly labeled in order to identify correctly labeled *bridges*.

Intuitively, informativeness of an instance can be measured by the prediction results of the ideal classifier $IS$ for the source domain and the ideal classifier $IT$ for the target domain. If $IS$ and $IT$ are both probabilistic classifiers, $IS$ should return a noninformative distribution while $IT$ should return an informative one. The ideal classifier for the source domain is approximated with a *source classifier* $SC$ trained on $D_S$, while the ideal classifier for the target domain is approximated by training a classifier, $TC$, on target domain instances labeled by the source classifier.

We also try to ensure that instances that are selected are correctly classified. As the label used is provided by the target classifier, we estimate the precision of the target classification. The final ranking function is constructed by combining this estimate with the informativeness of the instance.

We show the algorithm for the instance selection in Algorithm 2. The notations used follow those used in Algorithm 1. For simplicity, we assume that $w_t^S = w_t^T = 1$ for all $t$. We expect $TC$ to be a reasonable classifier on $D_T$ due to the presence of *generalists* and *bridges*. Note that the target classifier is constructed by randomly splitting $D_T$ into two partitions, training a classifier on each partition and using the prediction of the trained classifier on the partition it is not trained on. This is because classifiers tend to fit the data that they have been trained on too well making the probability estimates on their training data unreliable. Also, a random partition is used to ensure that the data in each partition is representative of $D_u$.

## 3.3 The scoring function: $score(p^{(s)}, p^{(t)})$

The scoring function $score(p^{(s)}, p^{(t)})$ in Algorithm 2 is simply implemented as the product of two components: a measure of the informativeness and the probability that $SC$'s label is correct. We show how the intuitive ideas (described above) behind these two components are formalized.

Informativeness of a distribution $p$ on a set of discrete labels $Y$ is measured by its entropy $h(p)$ defined by

$$h(p) = -\sum_{y \in Y} p(y) \log p(y).$$

**Algorithm 2** Algorithm for selecting instances for bootstrapping at iteration $t$

```
Input: Labeled source domain data $D_S$, target domain training data $T_t$, remaining data $R_t$, the classifier $SC_t$ trained on $D_S \cup T_t$, and a scoring function $score(p^{(s)}, p^{(t)})$
Output: $k$ instances for bootstrapping.

1. Label $R_t$ with $SC_t$, and to each instance $x_i \in R_t$, $SC_t$ outputs a distribution $p_i^{(s)}(y_i)$ over its labels.
2. Randomly split $R_t$ into two partitions, $R_t^0$ and $R_t^1$ with their labels assigned by $SC_t$.
3. Train each target classifier, $TC_t^x$ with the data $R_t^x$, for $x = \{0, 1\}$.
4. Label $R_t^{(1-x)}$ with the classifier $TC_t^x$, which to each instance $x_i \in R_t$, outputs a distribution $p_i^{(t)}(y_i)$ over its labels.
5. Score each instance from $x_i \in R_t$ with the function $score(p_i^{(s)}, p_i^{(t)})$.
6. Select top $k$ instances from $R_t$ with the highest scores.
```

$h(p)$ is nonnegative; $h(p) = 0$ if and only if $p$ has probability 1 on one of the labels; $h(p)$ attains its maximum value when the distribution $p$ is uniform over all labels. Hence, an instance is classified with high confidence when the distribution over its labels has low entropy.

We measure the informativeness of an instance using $h(p^{(s)}) - h(p^{(t)})$, where $p^{(s)}$ and $p^{(t)}$ are as in Algorithm 2. We argue that a larger value of this expression implies that the instance is more likely to be a *bridge* instance. This expression has a high value when the source classifier is uncertain, and the target classifier is certain. Uncertain classification by the source classifier indicates that the instance is unlikely to be a *generalist*. Moreover, if the target classifier is certain on $x_i$, it means that instances similar to the instance $x_i$ are consistently labeled with the same label by the source classifier $SC_t$, indicating that it is likely to be a *bridge* instance.

The probability that $TC$'s label is correct cannot be estimated directly because we do not have labeled target domain data. Instead, we use the source domain to give an estimate. We do this with a simple pre-processing step: we split the data $D_S$ into two partitions of equal size, train a classifier on each partition, and test each classifier on the other partition. We then measure the resulting accuracy given each label:

$$\rho(y) = \frac{\text{\# correctly labeled instances of label } y}{\text{\# total instances of label } y}.$$

Summarizing the above discussion, the scoring function is as shown below.

$$score(p^{(s)}, p^{(t)}) = \rho(y^*) \left[ h(p^{(s)}) - h(p^{(t)}) \right],$$
$$\text{where } y^* = \arg\max_{y \in Y} p^{(s)}(y)$$

The scoring function has a high value when the information content of the example is high and the label has high precision.

### 3.4 The termination criterion

Intuitively, our algorithm terminates when there are not enough informative instances. Formally, we define the termination criterion as follows: we terminate the bootstrapping process when, there exists an instance $x_i$ in the top $k$ instances satisfying the following condition:

1. $h(p_i^{(s)}) < h(p_i^{(t)})$, or
2. $\max_{y \in Y} p_i^{(s)}(y) > \max_{y \in Y} p_i^{(t)}(y)$

The second case is used to check for instances where the classifier $SC_t$ is more confident than the target classifiers $TC_t^x$, on their respective predicted labels. This shows that the instance $x_i$ is more of a *generalist* than a *bridge*.

## 4 NER task and implementation

The algorithm described in Section 3 is not specific to any particular application. In this paper, we apply it to the problem of named entity recognition (NER). In this section, we describe the NER classifier and the features used in our experiments.

### 4.1 NER features

We used the features generated by the CRF package (Finkel et al., 2005). These features include the word string feature, the case feature for the current word, the context words for the current word and their cases, the presence in dictionaries for the current word, the position of the current word in the sentence, prefix and suffix of the current word as well as the case information of the multiple occurrences of the current word. We use the same set of features for all classifiers used in the bootstrapping process, and for all baselines used in the experimental section.

### 4.2 Machine learning algorithms

A base machine learning algorithm is required in bootstrapping approaches. We describe the two machine learning algorithms used in this paper. We chose these algorithms for their good performance on the NER task.

**Maximum entropy classification (MaxEnt):** The MaxEnt approach, or logistic regression, is one of the most competitive methods for named entity recognition (Tjong and Meulder, 2003). MaxEnt is a discriminative method that learns a distribution, $p(y_i|x_i)$, over the labels, $y_i$, given the vector of features, $x_i$. We used the implementation of MaxEnt classifier described in (Manning and Klein, 2003). For NER, each instance represents a single word token within a sentence, with the feature vector $x_i$ derived from the sentence as described in the previous section. MaxEnt is not designed for sequence classification. To deal with sequences, each name-class (e.g. PERSON) is divided into sub-classes: first token (e.g. PERSON-begin), unique token (e.g. PERSON-unique), or subsequent tokens (e.g. PERSON-continue) in the name-class. To ensure that the results returned by MaxEnt is coherent, we define deterministic transition probabilities that disallow transitions such as one from PERSON-begin to LOCATION-continue. A Viterbi parse is used to find the *valid* sequence of name-classes with the highest probability.

**Support vector machines (SVM):** The basic idea behind SVM for binary classification problems is to consider the data points in their feature space, and to separate the two classes with a hyper-plane, by maximizing the shortest distance between the data points and the hyper-plane. If there exists no hyperplane that can split the two labels, the soft margin version of SVM will choose a hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples (Joachims, 2002). We used the SVM$^{light}$ package for our experiments (Joachims, 2002). For the multi-label NER classification with $N$ classes, we learn $N$ SVM classifiers, and use a softmax function to obtain the distribution. Formally, denoting by $s(y)$ the confidence returned by the classifier for each label $y \in Y$, the probability of the label $y_i$ is given by

$$p(y_i|x_i) = \frac{\exp(s(y_i))}{\sum_{y \in Y} \exp(s(y))}$$

Similarly to MaxEnt, we subdivide name-classes into begin, continue, and unique sub-classes, and use a Viterbi parse for the sequence of highest probability. The SVM$^{light}$ package also implements a transductive version of the SVM algorithm. We also compare our approach with the transductive SVM (Joachims, 1999) in our experimental results.

## 5 Experimental results

In this paper, we use the annotated data provided by the Automatic Content Extraction (ACE) program. The ACE data set is annotated for an Entity Detection task, and the annotation consists of the labeling of entity names (e.g. *Powell*) and mentions for each entity (e.g. pronouns such as *he*). In this paper, we are interested in the problem of recognition of the proper names (the named entity recognition task), and hence use only entities labeled with the type *NAM* (LDC, 2005). Entities are classified into seven types: *Person* entities are humans mentioned in a document; *Organization* entities are limited to established associations of people; *Geo-political* entities are geographical areas defined by political and/or social groups; *Location* entities are geographical items like landmasses and bodies of water; *Facility* entities refer to buildings and real estate improvements; *Vehicle* entities are devices used for transportation; and *Weapon* entities are devices used for harming or destruction.

We compare performances of a few algorithms: MaxEnt classifier (MaxEnt); MaxEnt classifier with standard bootstrapping (MaxEnt-SB); balanced bootstrapping based on MaxEnt classifier (MaxEnt-BB); MaxEnt with DAB (MaxEnt-DAB); SVM classifier (SVM); transductive SVM classifier (SVM-Trans); and DAB based on SVM classifier (SVM-DAB). No regularization is used for MaxEnt classifiers. SVM classifiers use a value of 10 for parameter C (trade-off between training error and margin). Bootstrapping based algorithms are run for 30 iterations and 100 instances are selected in every iteration.

The evaluation measure used is the F-measure. F-measure is the harmonic mean of precision and recall, and is commonly used to evaluate NER systems. We use the scorer for CONLL 2003 shared task (Tjong and Meulder, 2003) where the F-measure is computed by averaging F-measures for name-classes, weighted by the number of occurrences.

| **Code** | **Source** | **Num docs** |
|---|---|---|
| NW | Newswire | 81 |
| BC | Broadcast conversation | 52 |
| WL | Weblog | 114 |
| CTS | Conversational Telephone Speech | 34 |

Table 1: The sources, and the number of documents in each source, in the ACE 2005 data set.

### 5.1 Cross-source transfer

The ACE 2005 data set consists of articles drawn from a variety of sources. We use the four categories shown in Table 1. Each category is considered to be a domain, and we consider each pair of categories as the source and the target domain in turn.

Figure 1 compares the performance of MaxEnt-SB, MaxEnt-BB and MaxEnt-DAB over multiple iterations. Figure 2 compares the performance of SVM, SVM-Trans and SVM-DAB. Each line in the figures represents the average F-measure across all the domains over many iterations. When the termination condition is met for one domain, its F-measure remains at the value of the final iteration.

Despite a large number of iterations, both standard and balanced bootstrapping fail to improve performance. Supervised learning performance on each domain is shown in Table 3 (by 2-fold cross-validation with random ordering) as a reference. In Table 5, we compare the F-measures obtained by different algorithms at the last iteration they were run. We will discuss more on this in Section 5.3.

### 5.2 Cross-topic transfer

This data set is constructed from 175 articles from the ACE 2005 corpus. The data set is used to evaluate transfer across topics. We manually classify the articles into 4 categories: military operations (MO), political relationship or politicians (POL), terrorism-related (TER), and those which are not in the above categories (OTH). A detailed breakdown of the number of documents in the each topic is given in Table 2.

Supervised learning performance on each domain is shown in Table 4 (by 2-fold cross-validation with random ordering) as a reference. Experimental results on cross-topic evaluation are shown in Table 6. Figure 3 compares the performance of MaxEnt-SB, MaxEnt-BB and MaxEnt-

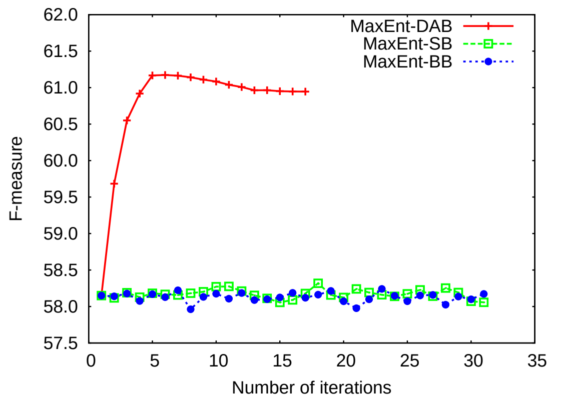

Figure 1: Average performance on the cross-source transfer using MaxEnt classifier.

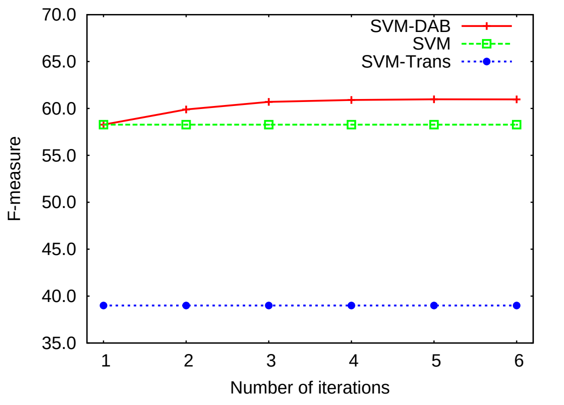

Figure 2: Average performance on the cross-source transfer using SVM classifier.

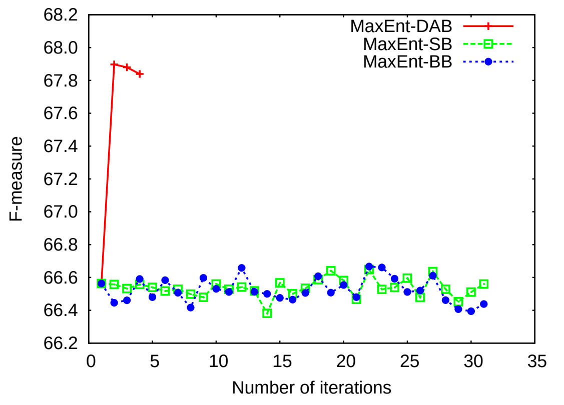

Figure 3: Average performance on the cross-topic transfer using MaxEnt classifier.

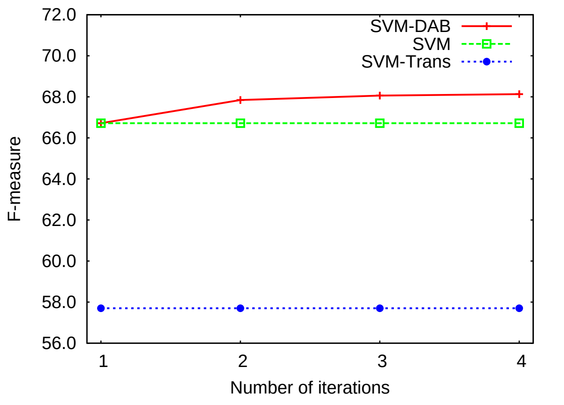

Figure 4: Average performance on the cross-topic transfer using SVM classifier.

| Topic | Topic description | # docs |
|---|---|---|
| MO | Military operations | 92 |
| POL | Political relationships | 40 |
| TER | Terrorist-related | 28 |
| OTH | None of the above | 15 |

Table 2: The topics, their descriptions, and the number of training and test documents in each topic.

| Domain | MaxEnt | SVM |
|---|---|---|
| MO | 80.52 | 80.6 |
| POL | 77.99 | 79.05 |
| TER | 81.74 | 82.12 |
| OTH | 71.33 | 72.08 |

Table 4: F-measure of supervised learning on the cross-topic target domains.

| Domain | MaxEnt | SVM |
|---|---|---|
| NW | 82.47 | 82.32 |
| BC | 78.21 | 77.91 |
| WL | 71.41 | 71.84 |
| CTS | 93.90 | 94.01 |

Table 3: F-measure of supervised learning on the cross-source target domains.

DAB over multiple iterations. Figure 4 compares the performance of SVM, SVM-Trans and SVM-DAB. Similar to cross-source transfer, standard and balanced bootstrapping perform badly. This will be discussed in Section 5.3.

## 5.3 Discussion

We show in our experiments that DAB outperforms standard and balanced bootstrapping, as well as the transductive SVM. We have also shown DAB to be robust across two state-of-the-art classifiers, MaxEnt and SVM. Balanced bootstrapping has been shown to be more effective for domain adaptation than standard bootstrapping (Jiang and Zhai, 2007) for named entity classification on a subset of the dataset used here. In contrast, we found that both methods perform poorly on domain adaptation for NER. In named entity classification, the names have already been segmented out and only need to be classified with the appropriate class. However, for NER, the names also

| Train | Test | MaxEnt | MaxEnt-SB | MaxEnt-BB | MaxEnt-DAB | SVM | SVM-Trans | SVM-DAB |
|---|---|---|---|---|---|---|---|---|
| BC | CTS | 74.26 | 74.19 | 74.16 | **81.03** | 72.47 | 43.27 | 75.43 |
| BC | NW | 64.81 | 64.76 | 64.80 | **66.20** | 64.08 | 43.01 | 64.39 |
| BC | WL | 47.81 | 47.80 | 47.76 | **49.52** | 47.98 | 36.58 | 47.93 |
| CTS | BC | 46.19 | 46.12 | 46.40 | **54.62** | 46.02 | 40.44 | 49.64 |
| CTS | NW | 54.25 | 54.15 | 54.26 | 53.07 | 55.63 | 23.61 | **58.99** |
| CTS | WL | 40.42 | 40.43 | 40.72 | 41.27 | 39.96 | 29.05 | **42.04** |
| NW | BC | 59.90 | 59.83 | 59.80 | **60.55** | 59.89 | 45.71 | 58.42 |
| NW | CTS | 66.64 | 66.48 | 66.59 | 66.73 | 68.28 | 28.80 | **73.47** |
| NW | WL | 52.52 | 52.53 | 52.47 | **53.44** | 52.19 | 36.39 | 52.30 |
| WL | BC | 58.58 | **58.79** | 58.65 | 56.00 | 58.43 | 52.64 | 58.64 |
| WL | CTS | 64.63 | 63.89 | 64.50 | 80.45 | 65.96 | 45.04 | **81.04** |
| WL | NW | 67.79 | 67.72 | 67.92 | 68.46 | 68.38 | 43.40 | **69.33** |
| Average | | 58.15 | 58.06 | 58.17 | 60.95 | 58.27 | 39.00 | **60.97** |

Table 5: F-measure of the cross-source transfer.

| Train | Test | MaxEnt | MaxEnt-SB | MaxEnt-BB | MaxEnt-DAB | SVM | SVM-Trans | SVM-DAB |
|---|---|---|---|---|---|---|---|---|
| MO | OTH | 81.70 | 81.48 | 81.57 | **81.95** | 81.78 | 75.68 | 81.94 |
| MO | POL | 73.21 | 73.11 | 73.28 | **74.97** | 72.56 | 58.13 | 72.66 |
| MO | TER | 68.13 | 68.07 | 68.24 | **69.89** | 69.40 | 65.02 | 69.38 |
| OTH | MO | 63.30 | 63.80 | 63.94 | 63.91 | 64.18 | 61.03 | **65.45** |
| OTH | POL | 67.96 | 68.05 | 67.86 | 69.13 | 68.29 | 56.50 | **70.67** |
| OTH | TER | 45.34 | 44.82 | 45.30 | 51.06 | 45.71 | 48.77 | **52.87** |
| POL | MO | 62.14 | 62.12 | 61.95 | 61.94 | 61.98 | 51.67 | **62.32** |
| POL | OTH | 77.91 | 77.72 | 77.79 | 76.58 | 78.11 | 65.71 | **78.13** |
| POL | TER | 66.55 | 66.38 | 66.08 | 66.38 | 66.44 | 51.29 | **67.24** |
| TER | MO | 58.35 | **58.62** | 58.02 | 57.29 | 58.30 | 49.80 | 58.14 |
| TER | OTH | 66.83 | 67.61 | 66.83 | **68.97** | 66.28 | 58.25 | 68.12 |
| TER | POL | 67.34 | 66.94 | 67.16 | **72.00** | 67.54 | 50.55 | 70.65 |
| Average | | 66.56 | 66.56 | 66.50 | 67.84 | 66.71 | 57.70 | **68.13** |

Table 6: F-measure of the cross-topic transfer.

need to be separated from not-a-name instances. We find that the addition of not-a-name instances changes the problem - the not-a-names form most of the instances classified with high confidence. As a result, we find that both standard and balanced bootstrapping fail to improve performance: the selection of the most confident instances no longer provide sufficient new information to improve performance.

We also find that transductive SVM performs poorly on this task. This is because it assumes that the unlabeled data comes from the same distribution as the labeled data. In general, applying semi-supervised learning methods directly to [S+T-] type domain adaptation problems do not work and appropriate modifications need to be made to the methods.

The ACE 2005 data set also contains a set of ariticles from the *broadcast news* (BN) source which is written entirely in lower case. This makes NER much more difficult. However, when BN is the source domain, the capitalization information can be discovered by DAB. Figures 5 and 6 show the average performance when BN is used as the source domain and all other domains in Table 1 as the target domains.

The source domain classifier tends to have high precision and low recall, DAB results in an increase in recall, with a small decrease in precision.

Testing the significance of the F-measure is not trivial because the named entities wrongly labeled by two classifiers are not directly comparable. We tested the labeling disagreements instead, using a McNemar paired test. The significance test is performed on the improvement of MaxEnt-DAB over MaxEnt and SVM-DAB over SVM. In most of the domains for the cross-source transfer, the improvements are significant at a significance level of 0.05, using MaxEnt classifier. The exceptional train-test pairs are NW-WL and WL-BC. In the case of WL-BC, this means the slight decrement in performance is not statistically significant. Similar result is achieved for the cross-source transfer using SVM classifier. In the cross-topic transfer, the source domain and the target domain are not very different. When we have a large amount of training data and little testing data, the gain of DAB can be not statistically significant, as in the case when we train with MO and POL domains.

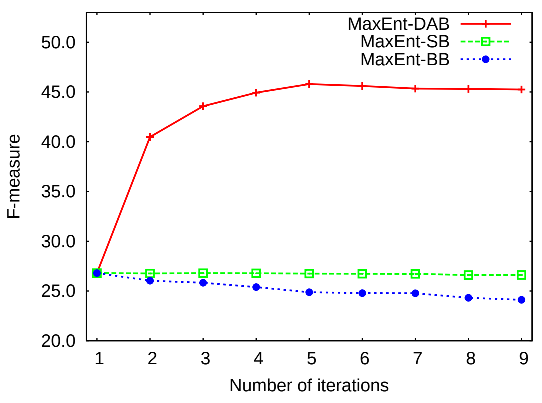

Figure 5: Performance on recovering capitalization using MaxEnt classifier.

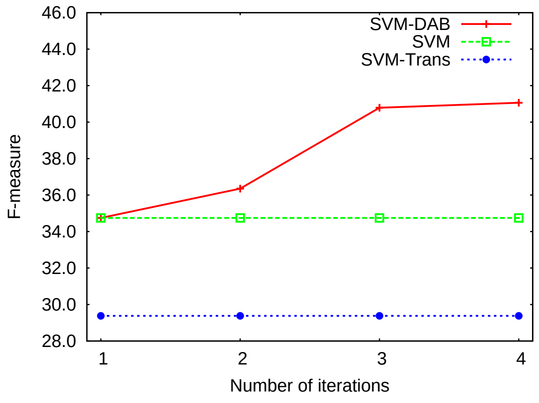

Figure 6: Performance on recovering capitalization using SVM classifier.

## 6 Conclusion

We proposed a bootstrapping approach for domain adaptation, and we applied it to the named entity recognition task. Our approach leverages on instances that serve as *bridges* between the source and target domain. Empirically, our method outperforms baseline approaches including supervised, transductive and standard bootstrapping approaches. It also outperforms balanced bootstrapping, an approach designed for domain adaptation (Jiang and Zhai, 2007).

## References

Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 81–88, New York, NY, USA. ACM Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 49–56, Prague, Czech Republic, June. Association for Computational Linguistics.

Massimiliano Ciaramita and Yasemin Altun. 2005. Named-entity recognition in novel domains with external lexical knowledge. In *Advances in Structured Learning for Text and Speech Processing Workshop*.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, New York City, USA. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA. Association for Computational Linguistics.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Linguistic Data Consortium LDC. 2005. ACE (Automatic Content Extraction) English Annotation Guidelines for Entities.

Christopher Manning and Dan Klein. 2003. Optimization, maxent models, and conditional estimation without magic. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on*

*Human Language Technology*, pages 8–8, Morristown, NJ, USA. Association for Computational Linguistics.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 589–596, Barcelona, Spain, July.

Erik Tjong and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of Conference on Computational Natural Language Learning*.